Multiple Factors-Aware Diffusion in Social Networks

Chung-Kuang Chou(✉) and Ming-Syan Chen

Department of Electrical Engineering, National Taiwan University, Taipei, Taiwan
ckchou@arbor.ee.ntu.edu.tw, mschen@cc.ee.ntu.edu.tw

Abstract. Information diffusion is a natural phenomenon that information propagates from nodes to nodes over a social network. The behavior that a node adopts an information piece in a social network can be affected by different factors. Previously, many diffusion models are proposed to consider one or several fixed factors. The factors affecting the adoption decision of a node are different from one to another and may not be seen before. For a different scenario of diffusion with new factors, previous diffusion models may not model the diffusion well, or are not applicable at all. In this work, our aim is to design a diffusion model in which factors considered are flexible to extend and change. We further propose a framework of learning parameters of the model, which is independent of factors considered. Therefore, with different factors, our diffusion model can be adapted to more scenarios of diffusion without requiring the modification of the diffusion model and the learning framework. In the experiment, we show that our diffusion model is very effective on the task of activation prediction on a Twitter dataset.

Keywords: Social networks · Diffusion models

1 Introduction

Information diffusion in social networks has been an active research field in about a decade. It is a natural phenomenon that information propagates from nodes to nodes over a social network, which acts like an epidemic. There are many applications on information diffusion, such as promoting an idea more effectively [5,10] , blocking adverse opinions [3,11], or identifying information flows [13] in a network. A well-known problem therein is called *influence maximization*, formulated by Kempe et al. [10]. The problem of influence maximization is to find a group of target nodes to be convinced of an idea initially to maximize the spread size, i.e. the number of nodes adopting the idea, on a given diffusion model.

To model how information diffuses in a network, researchers have proposed various diffusion models from different aspects. In these diffusion models, the Independent Cascading (IC) model and the Linear Threshold (LT) Model [10]

T. Cao et al. (Eds.): PAKDD 2015, Part I, LNAI 9077, pp. 70–81, 2015.
DOI: 10.1007/978-3-319-18038-0_6

have been widely employed in many applications and have several variants [1]. Both models consider the influence strength of a neighbor. The key difference is that, for a node turning to adopt an idea, IC considers only the influence from exactly one activated neighbor with uncertainty, while LT considers the collaborative influence contribution from all activated neighbors. In other words, IC places importance on which neighbor tries to affect the node to adopt the idea whereas LT thinks highly on the overall influence contribution from neighbors. Nevertheless, the real world is so complicated that a simple concern is hard to capture such complexity. Many factors probably affect the decision of adoption. For example, an idea that has been adopted by most people will have more chance to influence somebody [9] and an idea is harder to be adopted by someone as time passes. Moreover, a person may have different strength of interests in different topics [1]. However, the factors considered by previous diffusion models in social networks are all fixed. For a different scenario of diffusion with new factors, previous diffusion models may not model the diffusion well, or are not applicable at all. Therefore, one usually has to propose a new diffusion model for modeling diffusion of a specific scenario better by considering new factors.

To design a diffusion model for different factors one by one and to propose the corresponding algorithms, e.g. parameters learning and influence maximization, both become tedious. In this work, we aim to design a diffusion model which can consider multiple factors flexibly and further propose a framework of learning parameters of the model, related to information transmission likelihood between nodes and adoption prediction of a node. To the best of our knowledge, no existing work has the same sight. Specifically, we propose a Multiple-Factors Aware Diffusion (MFAD) model which is able to consider multiple factors flexibly that may affect adoption behaviors. MFAD is a two-stage propagation model. In the first stage, called *influence transmission*, an activated node u tries to influence its inactivated neighbor v with a probability. If the influence of u is successfully transmitted to v, in the second stage, called *adoption decision*, v decides whether it becomes activated based on its considerations, predicted via its related classification model trained on historical adoption information. Unfortunately, due to the limitation of observation in the real world, only positive instances are available to train classifiers , which is hard to achieve good performance. We further design a mechanism to get unlabeled instances to help train nodes' classifiers and propose the learning framework to learn the classifiers and transmission probabilities between nodes. Our contributions are summarized as follows.

1. Our proposed MFAD model is flexible to extend and change factors since we employ a classification approach to predicting the adoption behavior of a node.
2. Our proposed learning framework is independent of factors considered and we show the learning framework is effective in the experiment.
3. Due to the limitation of observation on diffusion in the real world, to predict adoption behaviors is hard to reach good results. We explicitly tackle this issue by learning nodes' classifiers for adoption decision with only positive and unlabeled instances.

The remaining of the paper is organized as follows. We next review the related work in Section 2. We introduce our diffusion model in Section 3 and how to learn the model in Section 4. In Section 5, we conduct experiments on a Twitter dataset. Finally, we conclude in Section 6.

2 Related Work

In this section, we briefly review the related works on diffusion models in social networks and learning parameters of diffusion models.

Diffusion models interpret how information spreads within a network. As mentioned above, IC and LT are two classical models and have been widely employed since they were connected to the influence maximization problem [10]. Recently, more factors of diffusion are explored in the literature like [1], [9] and [14], to name a few. N. Barbieri et al. [1] extend the IC model to consider topic distribution of items. T.-A. Hoang and E.-P. Lim [9] propose a model considering three factors, user virality, user susceptibility and item virality. Moreover, S.A. Myers et al. [14] explore not only internal influence from activated nodes in a network, but also external influence outside the network. However, as discussed above, the factors considered by these diffusion models are all fixed. For a different scenario of diffusion with new factors, these models may not model the diffusion well, or are not applicable at all.

Although diffusion models in social networks have been proposed for a long time, algorithms to learn parameters of a diffusion model are proposed recently. For example, K. Saito et al. [16] first propose a learning method for the IC model. The following up works mainly propose learning methods for their own diffusion models [1]. Moreover, A. Goyal et al. [7] propose the Credit Distribution model that directly estimates spread size from diffusion data without learning influence probabilities between nodes. Since we aim to design a learning framework that is independent of the diffusion factors considered, the above results do not apply to our scenario.

3 Proposed Model

In the work, our aim is to design a diffusion model which considers multiple factors flexibly for information propagation. We propose the Multiple Factors-Aware Diffusion (MFAD) model in the section.

Given a social graph $G = (V, E)$, where V is the node set and E is the edge set composed of directed edges without multiple edges and self-loops, let $p_{u,v}$ denote the probability of node u to successfully transmit influence to node $v \in u$'s out-neighbors $N^{out}(u)$ after u is activated by an item i and let $f_v(x)$ denote a probabilistic classifier for node v where $f_v(x)$ considers multiple predefined features, i.e. factors, that affect the tendency of u to adopt item i after v is exposed to i and x is the feature vector of the exposure. Note that a nonprobabilistic classification model, the outputs of which can be transformed to the probabilistic outputs [15], is applicable to $f_v(x)$, e.g. SVM with Platt scaling. The Multiple Factors-Aware Diffusion (MFAD) model is defined as follows.

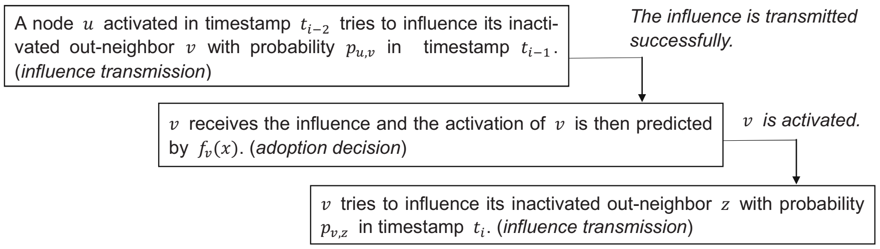

A node u activated in timestamp t_{i-2} tries to influence its inactivated out-neighbor v with probability $p_{u,v}$ in timestamp t_{i-1}. (*influence transmission*)
The influence is transmitted successfully.
v receives the influence and the activation of v is then predicted by $f_v(x)$. (*adoption decision*)
v *is activated.*
v tries to influence its inactivated out-neighbor z with probability $p_{v,z}$ in timestamp t_i. (*influence transmission*)

Fig. 1. The successful activation process of MFAD

Definition 1. *Multiple Factors-Aware Diffusion (MFAD) model. The propagation of diffusion starts with initial seed nodes $S_0 \subseteq V$ which have adopted item i. In the first timestamp t_1, each $u \in S_0$ tries to influence u's out-neighbors which are not activated by item i. The successful activation probability for $v \in N^{out}(u)$ is calculated as $p_{u,v}f_v(x)$. The activated nodes in t_1 are denoted as S_1. In timestamp t_2, new activated nodes in S_1 try to activate their inactivated out-neighbors in the same process as the above. The process runs iteratively. If S_j is empty in timestamp t_j, the diffusion terminates. Note that the spread size can be expressed as $| \cup_{0 \le i \le j-1} S_i|$ and when a node becomes activated, it never turns to be inactivated.*

MFAD is a two-stage propagation model. An illustration of the successful activation process in MFAD is shown as Figure 1. In the first stage, called *influence transmission*, a node u tries to influence each $v \in N^{out}(u)$ which is not activated with probability $p_{u,v}$ in timestamp t_{i-1}, where u is activated in timestamp t_{i-2}. However, the influence successfully transmitted over the edge does not directly activate a node. Our model has the following stage, called *adoption decision*. In the second stage, if the influence of u is successfully transmitted with probability $p_{u,v}$ to v previously, v receives this influence and decides whether it becomes activated based on its considerations, predicted via its classification model $f_v(x)$. If v is activated, v will try to influence its neighbors in the next timestamp t_i. Consider a news diffusing in an online social network, e.g. Facebook and Twitter. A user u posted a message about the news recently. Due to the ranking mechanism designed by Facebook or since there are too many messages, a friend v of u may not see the message, which is modeled by influence transmission. Moreover, even if the friend v reads the message, v may consider whether to share or reply to the message based on several concerns, e.g. v's interests, the importance of the news, which is modeled by adoption decision. In contrast to the traditional diffusion models, MFAD is flexible to consider multiple factors and considers more in a microscopic view for information propagation.

4 Two-Stage Learning

In this section, we propose a two-stage learning framework for MFAD since MFAD is a two-stage propagation model. In the first stage, the classifier of each node is trained, which corresponds to adoption decision, while the transmission probability between two connected nodes is estimated in the second stage, which corresponds to influence transmission. We first introduce the observed data.

Observed Data. A propagation trace consists of activation records. Each record $\{u, v, i, t\}$ represents that node u adopts an item i at timestamp t and the adoption is caused by u's in-neighbor v. If u is actually a seed of the item in the observed data, v does not exist and is set to be NIL. In reality, we usually do not observe that a node fails to influence others. In other words, we would only have positive instances for training classifiers directly from the propagation trace. We next discuss how to learn classifiers of nodes in such a situation.

4.1 Learning Classifiers of Nodes

Due to the limitation of observation in the real world, only positive instances are available to train binary classifiers of each nodes. However, a classifier trained on only positive instances is hard to achieve good performance. In fact, *unlabeled* instances can provide more information for learning, e.g. feature distribution, and can be generated via observing that an inactivated out-neighbor of an activated node does not turn activated in the next timestamp. We use the term *unlabeled* instead of *negative* since an unlabeled instance may be positive or negative due to the limitation of observation. Thus, the task of the first-stage learning becomes training classifiers by using positive and unlabeled instances. In the literature [6,12], the problem is called *positive and unlabeled learning*. Among previous work on positive and unlabeled learning, C. Elkan and K. Noto [6] provide a principled way to assigning weights to unlabeled instances. Based on their work [6], we construct a framework to learn nodes' classifiers for MFAD. In the following, we first describe how to obtain unlabeled instances from observed positive records and then describe how to train a node's classifier based on positive and unlabeled instances.

Obtaining Unlabeled Instances. With observed positive records, we can analyze the whole propagation trace to get positive instances for training nodes' classifiers with ease. However, negative instances are hard to obtain due to two main reasons: (1) an item does not successfully be exposed to a node from its in-neighbor; (2) the observation window for a node is not long enough. Fortunately, we can generate unlabeled instances to help train a node's classifier.

Assume that we have the complete propagation trace which consists of positive records in the format of $(u, s, i, t, o = 1)$ where the binary variable o indicates whether a record is an observed positive record in the trace or not. If a node u adopts an item i from s at timestamp t, we can observe u's out-neighbors who haven't adopted i from timestamp t. For an out-neighbor v of u, if v does not adopt i in the complete propagation trace, we generate an *unlabeled record* $(v, u, i, t', o = 0)$, where t' is the end observation time of propagation trace. However, the approach is with high cost and does not work if the size of the trace is extremely large. For a program to sequentially trace the positive records in

Algorithm 1. Unlabeled Record Generation

```
Require: graph $G = (V, E)$, complete propagation trace $\mathbb{D}$ sorted in chronological
    order, time window $T$
Ensure: unlabeled records
 1: Let $M$ be a table to memorize possible unlabeled instances
 2: for all $(u, s, i, t, 1)$ in $|\mathbb{D}|$ do
 3:     while $M$ contains a record $(u, *, i, *, 0)$ do
 4:         remove the record from $M$
 5:     for all $v$ in $u$'s out-neighbors do
 6:         if $v$ hasn't adopted $i$ then
 7:             insert an unlabeled record $(v, u, i, t, 0)$ into $M$
 8:     while the timestamp of the oldest record in $M$ is less than $t - T$ do
 9:         output and remove the record from $M$
10: while $|M| > 0$ do
11:     output and remove the record from $M$
```

chronological order, it has to memorize all items that a node has not adopted in order to generate unlabeled records at the end of tracing, which is impossible for a single machine with limited memory size. Otherwise, multiple scans are needed, which incurs many disk I/O operations and therefore is time-consuming.

In a more general way, we propose Algorithm 1 to trace positive records to generate unlabeled records. Let T be the time window to observe whether an out-neighbor v of u, for a positive record $(u, s, i, t, 1)$, adopts i before $t + T$. The main idea of the algorithm is that if v does not adopt i before $t + T$, the algorithm generates an unlabeled record $(v, u, i, t, 0)$. Although the pseudo code of Algorithm 1 is written in a batch way, it is easy to adapt it to process positive records coming in a streaming way. Note that a feature vector x, i.e. an instance, is generated at the same time when a positive record is traced or an unlabeled record is generated in order to capture the state of an exposure.

Training a Node's Classifier. With the above approach, for a node u, we can obtain positive instances $\mathbb{P}(u)$ and unlabeled instances $\mathbb{U}(u)$ for training u's classifier in order to predict the adoption tendency of an instance. Given an instance x, the goal is to predict $p(a = 1|x)$, where a is a binary random variable to indicate whether the instance is positive ($a = 1$) or negative ($a = 0$). Recall that o is a binary variable to indicate an instance is observed ($o = 1$) or unlabeled ($o = 0$). In the lemma derived in [6], $p(a = 1|x) = p(o = 1|x)/c$, where $c = p(o = 1|a = 1)$ is a constant value[1]. Based on the lemma, they [6] further reach the result on how to give weights to instances rigorously as the following. The weight of a positive instance is still unit, while an unlabeled instance have two copies, where one copy is a positive instance with weight $p(a = 1|x, o = 0)$ and the other copy is a negative instance with weight $1 - p(a = 1|x, o = 0)$.

[1] Due to the space limit, we omit the details of the lemma. If readers are interested in the lemma and the corresponding results, please refer to [6].

Note that $p(a=1|x,o=0)=\frac{1-c}{c}\frac{p(o=1|x)}{1-p(o=1|x)}$. Thus, our task here becomes three subtasks: (1) to learn $p(o=1|x)$, (2) estimate c and (3) learn $p(a=1|x)$.

Specifically, for a node u, (1) we first train a nontraditional classifier $g_u(x)=p_u(o=1|x)$ on $\mathbb{P}(u)-\mathbb{V}(u)$ and $\mathbb{U}(u)$, where the instances in $\mathbb{V}(u)$ are randomly selected from $\mathbb{P}(u)$, which is reserved as a validation set to estimate c. (2) Next, c is estimated as $\frac{1}{|\mathbb{V}(u)|}\sum_{x\in\mathbb{V}(u)} g_u(x)$ according to [6]. (3) Finally, we construct positive instances $\mathbb{P}'(u)$ and negative instances $\mathbb{N}(u)$ to train a traditional classifier $f_u(x)=p_u(a=1|x)$ for the node u. $\mathbb{P}'(u)$ contains the instances in $\mathbb{P}(u)$ and copies from the instances in $\mathbb{U}(u)$ with each weight $p_u(a=1|x,o=0)$. $\mathbb{N}(u)$ consists of copies from the instances in $\mathbb{U}(u)$ with each weight $1-p_u(a=1|x,o=0)$. In the experiments, we use the logistic regression to train both $g_u(x)$ and $f_u(x)$ since its output probability is well-calibrated [6] by applying the above way.

4.2 Learning the Transmission Probability

With the above $\mathbb{P}(v)$, $\mathbb{U}(v)$ and the trained classifier $f_v(x)$ for each node $v\in V$, we now describe how to learn transmission probability between two connected nodes. Let $\mathbb{D}=\bigcup_{v\in V}(\mathbb{P}(v)\cup\mathbb{U}(v))$ denote the dataset for learning transmission probabilities. An instance $x\in\mathbb{D}$ is in the format of $(\mathsf{f}_1,\mathsf{f}_2,...,\mathsf{f}_m)_{[u,v,o,t,i]}$ where u is the node that tries to activate v by item i before time t ($x_o=1$) or at time t ($x_o=0$), o is a binary variable to indicate whether v is activated during the observation in Algorithm 1 and $\mathsf{f}_1,\mathsf{f}_2,...,\mathsf{f}_m$ are factors of adoption, calculated in the same time of running Algorithm 1 for the exposure. An instance $x\in\mathbb{D}$ is unlabeled if $x_o=0$; otherwise, x is positive.

We train the MFAD model via maximizing the likelihood of $\mathbb{D}$ in the MFAD model. Let D_s denote the data of node s in $\mathbb{D}$, i.e. $\mathcal{D}_s=\{x\in\mathbb{D}|x_v=s\}$, and let Θ denote all parameters of the MFAD model to learn, i.e. all transmission probabilities, and $\Theta_s=\{p_{q,s}|q\in N^{in}(s)\}$ denote transmission probabilities between node s and its in-neighbors $N^{in}(s)$. Assuming adoptions between nodes are independent, the complete data log-likelihood can be expressed as follows.

$$\mathcal{L}(\Theta;\mathbb{D})=\log\prod_{s\in V}\mathcal{L}(\Theta_s;\mathcal{D}_s) \tag{1}$$

Note that since we want to learn transmission probability between two connected nodes, we exclude an instance x, x_u of which is NIL, from $\mathbb{D}$. Moreover, since $f_s(x)$ for each node s has trained in the above, the data likelihood of each $\mathcal{D}_s$ is only related to Θ_s. To maximize Eq.(1) is equal to maximizing each $\mathcal{L}(\Theta_s;\mathcal{D}_s)$,i.e.

$$\forall s\in V,\max_{\Theta_s}\log\mathcal{L}(\Theta_s;\mathcal{D}_s). \tag{2}$$

In reality, the diffusion happens in a continuous time space, while MFAD is a discrete time-based diffusion model. We include time constraints Δ^+ and Δ^- to decide the validity of an instance. Let $\mathcal{D}^+=\{x\in\mathbb{D}|x_o=1\}$ and $\mathcal{D}^-=\{x\in\mathbb{D}|x_o=0\}$. We define $\mathcal{D}_s^+$ and $\mathcal{D}_s^-$ as follows.

$$\mathcal{D}_s^+=\{x\in\mathcal{D}_s|x_o=1\wedge\exists y\in\mathcal{D}^+(y_v=x_u\wedge y_i=x_i\wedge 0\le x_t-y_t\le\Delta^+)\} \tag{3}$$

$$\mathcal{D}_s^- = \{x \in \mathcal{D}_s | x_o = 0 \wedge \exists y \in \mathcal{D}^+ (y_v = x_u \wedge y_i = x_i \wedge 0 \leq x_t - y_t \leq \Delta^-)\} \quad (4)$$

Note that for an instance $x \in \mathcal{D}^+$, $x_u \in N^{in}(x_v)$ should hold, where $N^{in}(v)$ is v's in-neighbor set. The data likelihood of node s is then defined as

$$\mathcal{L}(\Theta_s; \mathcal{D}_s) = \prod_{q \in N^{in}(s)} \prod_{x \in \mathcal{D}_{q,s}^+} (p_{q,s} f_s(x)) \prod_{q \in N^{in}(s)} \prod_{x \in \mathcal{D}_{q,s}^-} (1 - p_{q,s} f_s(x)), \quad (5)$$

where

$$f_s(x) = p_s(a = 1|x), \mathcal{D}_{q,s}^+ = \{x \in \mathcal{D}_s^+ | x_u = q\} \text{ and } \mathcal{D}_{q,s}^- = \{x \in \mathcal{D}_s^- | x_u = q\}. \quad (6)$$

The data log-likelihood of node s is:

$$\log \mathcal{L}(\Theta_s; \mathcal{D}_s) = \sum_{q \in N^{in}(s)} [\sum_{x \in \mathcal{D}_{q,s}^+} \log(p_{q,s} f_s(x)) + \sum_{x \in \mathcal{D}_{q,s}^-} \log(1 - p_{q,s} f_s(x))] \quad (7)$$

To find $p_{q,s}$ by maximizing the above log likelihood, let $\frac{\partial \log \mathcal{L}(\Theta_s; \mathcal{D}_s)}{\partial p_{q,s}} = 0$:

$$\sum_{x \in \mathcal{D}_{q,s}^+} \frac{1}{p_{q,s}} + \sum_{x \in \mathcal{D}_{q,s}^-} \frac{-f_s(x)}{1 - p_{q,s} f_s(x)} = 0 \quad (8)$$

Since no closed form solution for Eq.(8) exists, we employ the Brent's algorithm [2]. The Brent's algorithm uses a combination of golden section search and successive parabolic interpolation. For an initial good guess $p_{q,s}^0$ in order to converge fast, we apply the first order Taylor series to approximate $\frac{-f_s(x)}{1 - p_{q,s} f_s(x)}$ at $p_{q,s} = 0$ as $-f_s(x) - f_s(x)^2 p_{q,s}$. Thus, the Eq. (8) becomes

$$\sum_{x \in \mathcal{D}_{q,s}^+} \frac{1}{p_{q,s}} + \sum_{x \in \mathcal{D}_{q,s}^-} [-f_s(x) - f_s(x)^2 p_{q,s}] = 0 \quad (9)$$

and by some mathematical manipulation we get $\hat{p}_{q,s} = \frac{-C + \sqrt{C^2 + 4BD}}{2D}$, where $B = |\mathcal{D}_{q,s}^+|$, $C = \sum_{x \in \mathcal{D}_{q,s}^-} f_s(x)$ and $D = \sum_{x \in \mathcal{D}_{q,s}^-} f_s(x)^2$. Note that D should be a real number greater than 0 and obviously, B and C are non-negative real numbers. In some situation, $\hat{p}_{q,s}$ will not be a valid probability value, the value $\frac{(3 - \sqrt{5})}{2}$ suggested in the Brent's algorithm [2] is used instead.

5 Experiments

In the section, we conduct experiments on a Twitter dataset to evaluate the effectiveness on activation prediction. We first describe the setup.

Table 1. Data Statistics

T	training instances				testing instances			
	positive	unlabeled		all	positive	unlabeled		all
		positive	negative			positive	negative	
3	17,197	274,575	236,807	528,579	115,955	80,243	15,288	211,486
6	17,197	270,467	236,807	524,471	114,463	78,788	15,940	209,191
12	17,197	263,750	236,807	517,754	112,063	76,496	17,001	205,560
24	17,197	252,274	236,807	506,278	108,180	72,864	18,893	199,937
48	17,197	233,820	236,807	487,824	101,010	66,049	20,705	187,764
96	17,197	204,919	236,807	458,923	88,807	54,793	24,520	168,120

5.1 Setup

Dataset. We use the real dataset collected from Twitter by L. Weng et al. [17]. We use standard preprocessing steps, similar to the steps used in [1] to clean diffusion data. However, in order to obtain enough size of training data for training nodes' classifiers, we allow multiple activation records of the same item for a node. After the preprocessing, each node has at least 20 activation records, i.e. retweets and tweets with hashtags in Twitter, and each item, i.e. hashtags, are adopted by at least 20 nodes. Moreover, there is no isolated node left. The remaining social graph consists of $24,045$ nodes and $871,745$ directed edges. The remaining activation records contain $8,427$ different items and the number of all activation records is $1,105,316$. The dataset spans from March 23 to April 25 in 2012, approximately one month long.

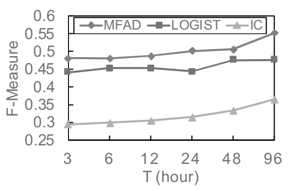
Factors. We first define some notations. Let $deg_{in}(v)$ and $deg_{out}(v)$ denote v's in-degree and out-degree in the graph. For an item i, we use $t_{glo}(i)$ to denote the earliest time in which i is adopted by some node in the data and $t_{loc}(i, v, t)$ to denote the earliest time in which i is exposed to v by an in-neighbor of v that adopts i before time t. Let $node_{glo}(i, t)$ denote all nodes activated by item i before time t and $node_{loc}(i, t, v)$ to represent in-neighbors of v which are activated by item i before time t. For a directed edge from u to v, we use $ratio_{from}(v, u, t)$ to denote the ratio that v's adoptions are caused by u and $ratio_{same}(v, u, t)$ to denote the ratio that v's adopted items are the same as u's adopted items before time t.

For an instance $x = (\mathsf{f}_1, \mathsf{f}_2, ..., \mathsf{f}_m)_{[u,v,o,t,i]}$, where node u tries to activate v by item i before time t ($x_o = 1$) or at time t ($x_o = 0$), the features $\mathsf{f}_1, \mathsf{f}_2, ..., \mathsf{f}_m$ are composed of three types, structure-based, time-based and history-based features. Structure-based features include $deg_{in}(u)$, $deg_{out}(u)$, $deg_{in}(v)$, $deg_{out}(v)$ and the number of common neighbors between u and v. Time-based features are $t - t_{glo}(i)$, $t - t_{loc}(i, v, t)$ and $t(i, u) - t$, where $t(i, u)$ is the time that node u adopts the item and if it is unavailable in the dataset, we assume $t(i, u) - t = 0$. The first two are able to reflect global and local freshness. The last one is to measure the adoption latency. History-based features are $|node_{glo}(i, t)|$, $|node_{loc}(i, t, v)|$, $ratio_{from}(v, u, t)$ and $ratio_{same}(v, u, t)$. Thus, there are $m = 12$ features in total for training a node's classifier in the experiment.

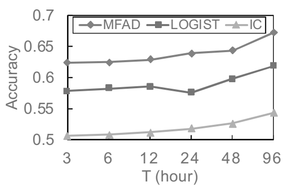
Instances. We use the approach introduced in Section 4.1 to generate positive and unlabeled instances from the dataset with different T for the most active 100 nodes, measured by the number of positive records in the whole dataset. The statistics of training and testing instances are summarized as Table 1. In both training and testing sets, we exclude instance x, x_u of which is NIL, since we want to learn transmission probability between two connected nodes for diffusion models. Note that the positive instances for testing consist of positive instances and unlabeled positive instances, while the negative instances for testing consist of unlabeled negative instances. For each T, we use the earliest 20% instances as the training set. From the latest 80% instances, the testing set only contains instances related to $p_{u,v}$ that is trained in the training data for MFAD. Thus, the satisfied testing instances are not too many. Moreover, the number of unlabeled instances for training is much more than the labeled positive instances since the earliest 20% time ($\sim$ 6.6 days) is relative short and when a node u adopts an item i at time $t(i,u)$ but u's in-neighbors all adopt i before $t(i,u) - T$, $|N^{in}(u)|$ unlabeled positive instances will be generated.

Methods. We include the following three methods to predict activations of nodes: (1) the logistic regression directly trained on positive and unlabeled instances (LOGIST), which is the classical approach, (2) our proposed learning framework for the MFAD model (MFAD) and (3) the independent cascading model (IC). Note that only MFAD and IC are diffusion models, while LOGIST is a classification algorithm only and cannot be applied to other applications on diffusion, e.g. influence maximization [10]. We select the IC model instead of the LT model since IC is also a probabilistic diffusion model. The parameters of IC are inferred by the maximum-likelihood estimation conducted in the same approach of Section 4.2. For two connected nodes u and v, the influence probability $p_{u,v}$ is $\frac{|\mathcal{D}^+_{u,v}|}{|\mathcal{D}^+_{u,v}|+|\mathcal{D}^-_{u,v}|}$ for IC. While training the nodes' classifiers for both LOGIST and MFAD, the class imbalance problem is encountered, i.e. skewed class distribution. We use SMOTE[4], which doubles the size of the minority class, and then apply SpreadSubsample [8] to undersample instances of the majority class to balance the class distribution. Moreover, we set time constraints Δ^+ and Δ^- in Eq. (3) and (4) as the same value of T. All methods are implemented in Java with Weka [8] and executed in a PC with an Intel i7 3.4GHz CPU. The running time of a run of MFAD for the same T does not exceed 2 hours, including time for sampling, training 100 nodes' classifiers and learning related transmission probabilities.

Metrics. We use four metrics, precision, recall, F-Measure and accuracy, to measure the results of activation prediction, based on true positive (TP), false positive (FP), true negative (TN) and false negative (FN) instances. Precision is $\frac{TP}{TP+FP}$. Recall is $\frac{TP}{TP+FN}$. F-Measure is $\frac{2 \times precision \times recall}{precision + recall}$ and accuracy is defined as $\frac{TP+TN}{TP+FP+TN+FN}$.

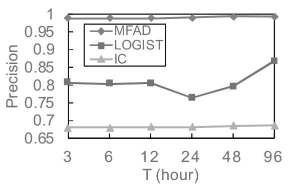


(a) F-Measure

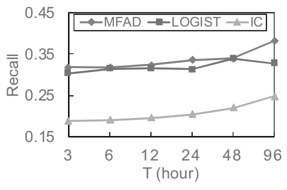


(b) Accuracy

Fig. 2. Overall Results



(a) Precision



(b) Recall

Fig. 3. Results of Components of F-Measure

5.2 Results

The overall results of activation prediction are shown in Figure 2. Our MFAD outperforms the other two methods significantly in the overall metrics, F-Measure and accuracy. F-Measure in Fig. 2(a) concerns mainly on true positive, false positive and false negative instances, while accuracy in Fig. 2(b) takes true negative instances into consideration. F-Measure is suitable for the scenario of retrieval of activated nodes whereas accuracy is more suitable for the scenario of spread estimation. MFAD works great for both scenarios. For the components of F-Measure, MFAD is very effective in precision as shown in Fig. 3(a), which means the size of false positive instances is much smaller than those of LOGIST and IC. The recalls of MFAD and LOGIST are close to each other but much better than that of IC as shown in Fig. 3(b). Moreover, as T increases, F-Measure and accuracy become better for all methods since the positive unlabeled instances are fewer and thus more positive instances are available for training classifiers.

In summary, MFAD is the best method to predict activation of nodes among three methods. Most importantly, MFAD is a diffusion model and therefore can simulate how information diffuses whereas LOGIST cannot. IC is also a diffusion model, but it cannot reflect the state of an exposure precisely and thus do not model the diffusion well. Although there is an extension of IC in [1], called TIC, to consider the topic factor, the dataset does not have the detailed textual information of tweets and hence we do not include TIC in the experiment. Nevertheless, our MFAD model can consider the topic factor by defining new features for nodes' classifiers with ease, which does not require the modifications of the learning framework.

6 Conclusions

In this work, we propose the model of Multiple-Factors Aware Diffusion Model (MFAD) which explicitly models *influence transmission* and *adoption decision* and considers multiple factors flexibly that may affect adoption behaviors. The learning framework of MFAD is independent of factors considered and is effective as shown in the experiment. Therefore, MFAD has more flexibility and can be applied to different scenarios for different purposes with ease. In the future, we will design influence maximization algorithms for MFAD.

Acknowledgments. This work is in part supported by MOST of Taiwan (103-2221-E-001-038-MY2).

References

1. Barbieri, N., Bonchi, F., Manco, G.: Topic-aware social influence propagation models. In: ICDM, pp. 81–90 (2012)
2. Brent, R.P.: Algorithms for minimization without derivatives. Prentice-Hall, Englewood Cliffs (1973)
3. Budak, C., Agrawal, D., El Abbadi, A.: Limiting the spread of misinformation in social networks. In: WWW, pp. 665–674 (2011)
4. Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, W.P.: Smote: Synthetic minority over-sampling technique. J. Artif. Int. Res. **16**(1), 321–357 (2002)
5. Chen, W., Wang, C., Wang, Y.: Scalable influence maximization for prevalent viral marketing in large-scale social networks. In: KDD, pp. 1029–1038 (2010)
6. Elkan, C., Noto, K.: Learning classifiers from only positive and unlabeled data. In: KDD, pp. 213–220 (2008)
7. Goyal, A., Bonchi, F., Lakshmanan, L.V.S.: A data-based approach to social influence maximization. Proc. VLDB Endow. **5**(1), 73–84 (2011)
8. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The weka data mining software: An update. SIGKDD Explor. Newsl. **11**(1), 10–18 (2009)
9. Hoang, T.A., Lim, E.P.: Virality and susceptibility in information diffusions. In: ISWSM (2012)
10. Kempe, D., Kleinberg, J., Tardos, E.: Maximizing the spread of influence through a social network. In: KDD, pp. 137–146 (2003)
11. Kimura, M., Saito, K., Motoda, H.: Blocking links to minimize contamination spread in a social network. ACM TKDD 3(2), 9:1–9:23 (2009)
12. Lee, W.S., Liu, B.: Learning with positive and unlabeled examples using weighted logistic regression. In: ICML (2003)
13. Lin, C., Mei, Q., Han, J., Jiang, Y., Danilevsky, M.: The joint inference of topic diffusion and evolution in social communities. In: ICDM, pp. 378–387 (2011)
14. Myers, S.A., Zhu, C., Leskovec, J.: Information diffusion and external influence in networks. In: KDD, pp. 33–41 (2012)
15. Niculescu-mizil, A., Caruana, R.: Predicting good probabilities with supervised learning. In: ICML, pp. 625–632 (2005)
16. Saito, K., Nakano, R., Kimura, M.: Prediction of information diffusion probabilities for independent cascade model. In: KES, pp. 67–75 (2008)
17. Weng, L., Menczer, F., Ahn, Y.Y.: Virality prediction and community structure in social networks. Nature Scientific Report 3(2522) (2013)